NODEJS

Training Assignment

Document Code	**25e-BM/HR/HDCV/FSOFT**
Version	**1.1**
Effective Date	**20/11/2012**

Hanoi, 12/2021

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1.	01/12/2021	Create a new Assignment	Create new		VinhNV

Contents

FSOFT ACADEMY	CODE	:	NodeJS_Training_Assignments
	TYPE	:	N/A
	LOC	:	N/A
	DURATION	:	<Duration in minutes>

Unit 2: RESTful APIs

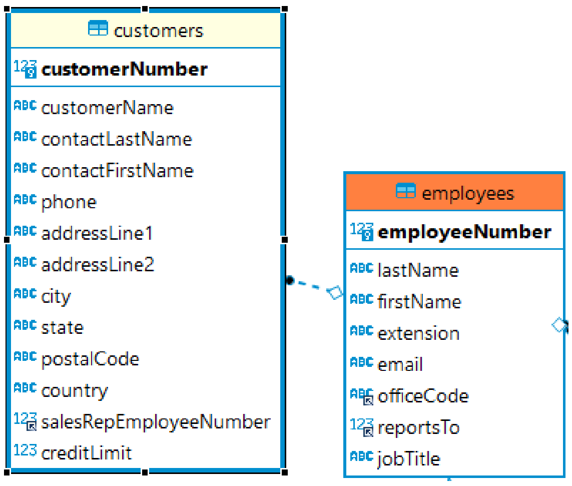
Objectives:

This assignment will help you familiarize with the basic concepts of REST and how to develop a set of RESTful APIs.

Problem Descriptions:

In this assignment, you will develop a set of REST APIs for a small company, capable of managing its employees, customers a server and returning information by using a collection of simple HTTP requests, as well as manipulating data for admin and manager of the company.

Technical Requirements:



- Attributes for each object is illustrated by the Entity Relation Diagram above
- Each customer is linked to an employee by salesRepEmployeeNumber
- Each employee CAN BE linked to another employee by reportsTo (optional)
- Your RESTful APIs should be able to respond, with the appropriate data for the following (but not limited to) routes:
  - /employees
  - /customers
- (Some routes might have multiple methods, and having custom query strings, params, request body,… it’s up to you to decide what is the best way to approach.)

- All CRUD operations are supported for those 2 resources.
- Each route managing its own respective data with appropriate HTTP method
- The data accepted and generated by the web service is encoded using JSON.
- Data stored can be managed by read/write files or database.

Estimated Time to complete: 240 mins

Unit 2: Authentication & Authorization

Problem Descriptions:

With the implementation of /employees & /customers endpoints, we are now able to access the resources on the server.

However, we are facing some problems with the services:

- Everyone with the endpoints can access our resources with the capabilities to update or delete the customers' sensitive information.
- Data input is not validated, not following any specific agreements or interfaces.
- When errors occur, no information is notified to the users, so they might not know what's wrong and how to fix it.
- Users not knowing what is needed for the requests.

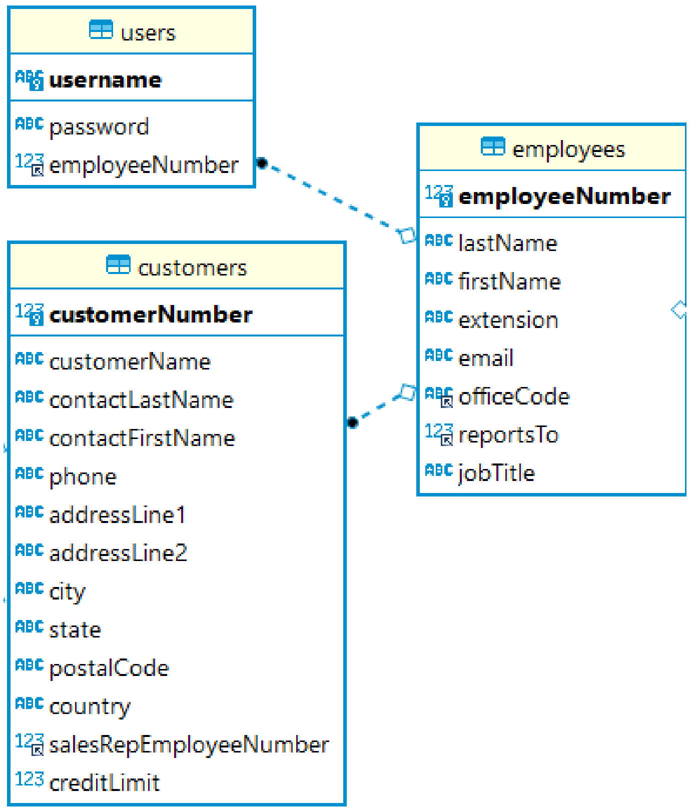
Step by step, we will tackle the problem one by one.

First, let's focus on the security vulnerabilities – giving access to only authorized personnel - authentication and authorization.

Assumptions:

- Day 6's Assignment is finished, with the implementation of /employees and /customers

Technical Requirements:



- Each employee can register for an account with simple information username – password
- Passwords stored in our system should be properly managed (hashed) using `bcrypt`
- Requests to the resources are managed using Authorization Bearer Token

- Token is generated and verified using jwt
- Token is properly signed and verified.
- Your RESTful APIs should be able to respond, with the appropriate data for the following (but not limited to) routes:
    - /employees
    - /customers
    - /users/register
    - /users/login
- (Some routes might have multiple methods, and having custom query strings, params, request body,… it's up to you to decide what is the best way to approach.)
- /employees and /customers also return linked data
- Only registered users can access the resources and endpoints other than /register & /login
- The access to the resources is listed for Roles in the Permission Matrix below:

Resource	Method	Roles			
		President	Manager	Leader	Staff
employees	Read	YES	YES	YES	NO
	Create	YES	YES	NO	NO
	Update	YES	YES	NO	NO
	Delete	YES	NO	NO	NO
customers	Read	YES	YES	YES (but only customer belong to employees in the same office)	YES (but only their own customers)
	Create	YES	YES	YES (but only customer belong to employees in the same office)	YES (but only customers belong to them)
	Update	YES	YES	YES (but only customer belong to employees in the same office)	NO
	Delete	YES	YES	YES (but only customer belong to employees in the same office)	NO

- Data stored can be managed by read/write files or database.

Estimated Time to complete: 300 mins

Unti 2: Data Modeling & Validation

Objectives:

- Understand the purposes of Data Modeling
- Data input is validated before handling further by our system

Problem Descriptions:

Tackling the next issue – validating the request input.

If the input is invalid, the request should be rejected and users getting error messages to know what went wrong

Assumptions:

- Day 6’s Assignment is finished, with the implementation of /employees and /customers
- Day 7’s Assignment is finished, with the implementation authentication & authorization

Technical Requirements:

- Requests are properly validated before further handling using `celebrate` & `Joi` (details for validation rules can be found below)
- The endpoints only accept pre-defined field, no additional fields.
- If any field is invalid, users should get errors notifying which field.
- Any field marked with `not able to update` should NOT be able to be updated when calling PUT/PATCH -> throwing 400 – error with message “{field} should not be changed.”

Endpoint	Method	Attribute	Details
employees	POST/PUT	employeeNumber	- required - positive number - not null - not able to update
		lastName	- required - string - min 3 chars – max 50 chars - not null - not able to update
		firstName	- required - string - min 3 chars – max 50 chars - not null

			- not able to update
		extension	- required - string - max 50 chars - not null
		email	- required - in an email format - min 10 chars – max 100 chars - not null
		officeCode	- required - string - max 10 chars - not null
		reportsTo	- optional - positive number - nullable
		jobTitle	- required - only allow to be one of the Role in the Permission Matrix
customers	POST/PUT	customerNumber	- required - positive number - not null - not able to update
		customerName	- required - string - min 5 chars – max 50 chars - not null
		contactLastName	- required - string - min 3 chars – max 50 chars - not null
		contactFirstName	- required

			- string - min 3 chars – max 50 chars - not null
		phone	- required - string - min 8 chars – max 20 chars - not null
		addressLine1	- required - string - min 10 chars – max 50 chars - not null
		addressLine2	- optional - string - min 10 chars – max 50 chars - nullable
		city	- required - string - min 2 chars – max 50 chars - not null
		state	- optional - string - min 2 chars – max 50 chars - nullable
		postalCode	- optional - string - min 5 chars – max 15 chars - nullable
		country	- required - string - min 2 chars – max 50 chars - not null
		salesRepEmployeeNumber	- required

			- positive number - nullable
		creditLimit	- optional - decimal (10, 2) - nullable
users/register		username	- required - string - min 3 chars – max 20 chars - not null
		password	- required - string - min 6 chars – max 100 chars - contains **at least** a number, a special char - not null
		employeeNumber	- required - positive number - not null

-

Estimated Time to complete: 240 mins

Unit 2: API Documentation

Start working with API Documentation – OpenAPI (Swagger)

Objectives:

Get familiarize with API Documentation

Problem Descriptions:

Users need to know how to use our APIs and what should they expect from our services.

API Documentation would provide needed information for users.

So in this assignment, let’s make sure our APIs sending the correct data to users then create an OpenAPI specification for our RESTful APIs and serve it as a web app for users to try out and get information related to the APIs.

Assumptions:

- Day 6’s Assignment is finished, with the implementation of /employees and /customers
- Day 7’s Assignment is finished, with the implementation authentication & authorization
- Day 8’s Assignment is finished, with the implementation data validation

Technical Requirements:

- Responses are properly validated using `Joi`. There should be no response sent to users in an unexpected format.
- All implemented API are documented using Swagger/OpenAPI
- Visualization of API Document can be accessed via /docs, with examples and are executable

Estimated Time to complete: 240 mins

Unit 2: Error Handling

Handle errors in our Application

Objectives:

- To understand the importance of logging and debugging
- To handle errors and exceptions in our system properly

Problem Descriptions:

In this assignment, we will implement the middleware needed to handle the errors or exceptions thrown when processing the request from users.

When the errors occurs, the errors should also be logged in a file for debugging later, so let's add a logger to our application.

Assumptions:

- Day 6's Assignment is finished, with the implementation of /employees and /customers
- Day 7's Assignment is finished, with the implementation authentication & authorization
- Day 8's Assignment is finished, with the implementation data validation

Technical Requirements:

- Requests/Responses and relevant data are logged into `info.log` file using Winston
- Errors are logged into `errors.log` using Winston
- Errors are handled using middlewares

Estimated Time to complete: xx mins

Unit 3: Working with Database

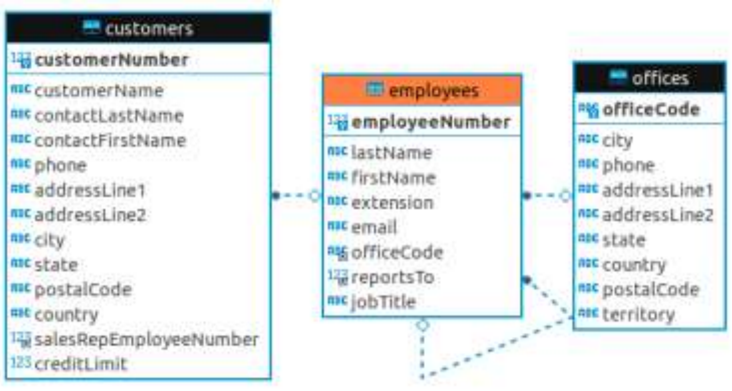
Basic Sql command

Objectives:

Help the learner familiar with sql language and know some basic sql command.

Problem Descriptions:

There is a logistic company with the database schema as



The company some reports about employees.

Assumptions:

- Trainees can import the exported mySql database provided by trainer.

Technical Requirements:

- Using mySql version 5.7

Questions to answer:

1. Update DB schema

- Create new table “Role”

id: Int(11) auto increase primary key

role: Varchar[50] not null unique

Insert values (President, Manager, Leader, Staff) with corresponding permissions defined in API session into the table.

- Add new column ‘role’ into employees table that references to ‘role’ table.

- Update all employees 'Sales Manager (APAC)', 'Sale Manager (EMEA)', 'Sales Manager (NA)' to ‘Manager’ role

- Update all employees who has person report to as |leader”

- Update all remaining employees , 'Sales Rep', 'VP Sales', 'VP Marketing' to ‘Staff’ role.

2. Write basic SQL commands

- Write CRUD commands for employees table.
- Report list of employee order by number of customers.
- Report list of offices order by number of customers.
- Report list of employee order by total credit limit of their customers.
- Report average customer’s credit limit of each office.

Estimated Time to complete: 300 mins

Unit 3: Working MySql with NodeJs

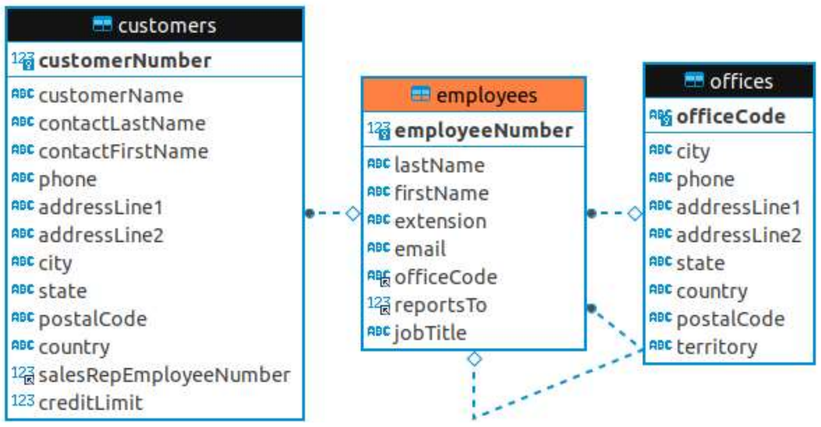
Using MySql with Nodejs

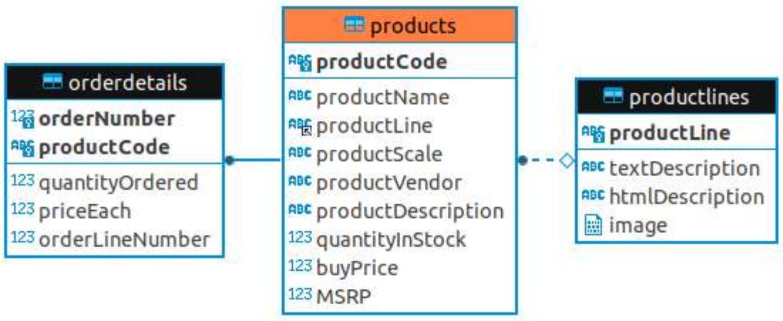
Objectives:

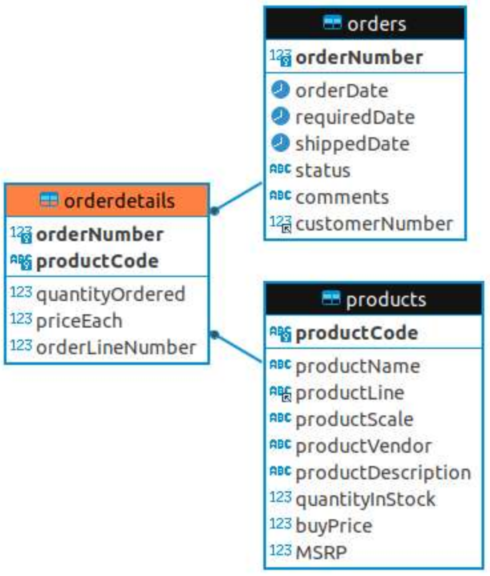
Help the learner familiar with ORM in NodeJs and know how to access relational DB.

Problem Descriptions:

There is a logistic company with the database schema as







The company needs some reports about employees.

Assumptions:

- Trainees finished the assignment of session 2.

- Trainees can import the exported mySql database provided by trainer.

Technical Requirements:

- Using mySql version 5.7

- Use Objection module.

- Using Express framwork.

Questions to answer:

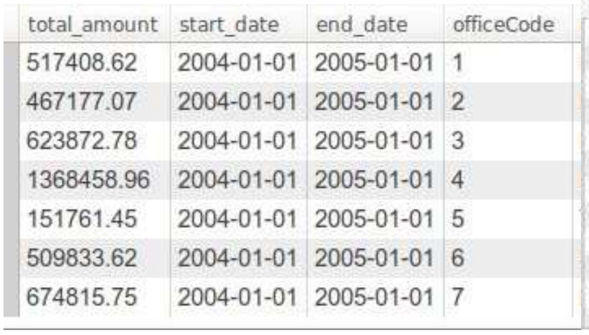
1. Create CRUD APIs that:

   - Allow to create, read data, update, delete a Customer.
   - Allow to create, read data, update, delete a Employee.

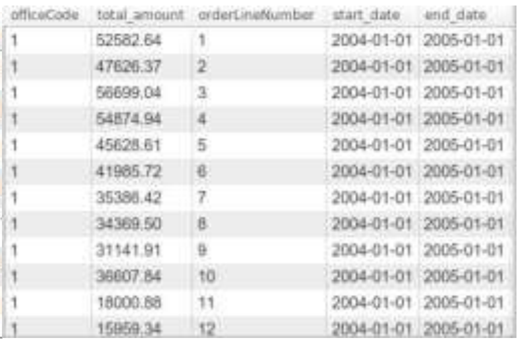
2. Validate all inputs in request. Ensure all responses to user are meaningfull.

3. (Advance)

   - In Api POST /employee, allow to input an array including customers. Create employee and all customers. Rejrect invalid customers (example customer name is empty) with meaningfull message.
   - With each office, create one default employee with last name as ‘9999’. This lastname is unique per office code and cannot be deleted. In Api /delete employee, move all of its associated customers to the default employee of the corresponding office code.
   - Create new Api Get /report that allow to get information:
     - Sale information of one or all offices in time period (with input start_date and end_date). Sale information include total revenue and revenue of each product line.
     - New customers of a input employee in a period with input start_date and end_date). Revenue from those customers.



total_amount	start_date	end_date	officeCode
517408.62	2004-01-01	2005-01-01	1
467177.07	2004-01-01	2005-01-01	2
623872.78	2004-01-01	2005-01-01	3
1368458.96	2004-01-01	2005-01-01	4
151761.45	2004-01-01	2005-01-01	5
509833.62	2004-01-01	2005-01-01	6
674815.75	2004-01-01	2005-01-01	7



officeCode	total_amount	orderLineNumber	start_date	end_date
1	52582.64	1	2004-01-01	2005-01-01
1	47626.37	2	2004-01-01	2005-01-01
1	56699.04	3	2004-01-01	2005-01-01
1	54874.94	4	2004-01-01	2005-01-01
1	45628.61	5	2004-01-01	2005-01-01
1	41985.72	6	2004-01-01	2005-01-01
1	35386.42	7	2004-01-01	2005-01-01
1	34369.50	8	2004-01-01	2005-01-01
1	31141.91	9	2004-01-01	2005-01-01
1	36607.84	10	2004-01-01	2005-01-01
1	18000.88	11	2004-01-01	2005-01-01
1	15959.34	12	2004-01-01	2005-01-01

Estimated Time to complete: 600 mins

Unit 3: Working MySql with NodeJs

NodeJs with MongoDB

Objectives:

Help the learner familiar with NoSQL database and know how to access to MongoDB in NodeJs.

Problem Descriptions:

System need store all log information in MongoDB.

Technical Requirements:

- Using Mongoose module

- Don’t use log module that support working with MongoDB natively (such as Winston)

Questions to answer:

1. Update Log into MongoDB

The log includes at least 4 information

- Log level: [“Error”, “Warning”, “Info”]
- User : User info who is performing the request.
- Message: Log content.
- CreatedAt: current time

- Error message log all information that is abnormal behaviour in system (normally exception in try catch).
- Warning: Invalid input data.
- Info: Information may help the developer debug the issue such as request content, query to DB, etc.

2. Write API for logs info that

- Update log level of the system.
- Get log info by level, user, time period, log content.

3. Additional requirement

- Write a feature simillar to logs, but it helps to tracking the performane of the system.

Estimated Time to complete: 150 mins

Unit 4: Unit testing, Integration testing and Code quality control

Assignment 1: The concept of Unit Testing - Integration Testing

Objectives:

Understanding about unit testing. Integration testing, and best practice.

Do the Quiz: (The bold is answer)

Quiz

1. What is Unit Testing?

A. Unit Testing (or component testing) refers to tests that verify the functionality of a specific section of code, usually at the function level.

B. In an object-oriented environment, this is usually at the class and methods level.

C. Unit Tests are written by the developers as part of the programming.

D. All the answers above.

2. Who is creating the Unit Tests?

A. The Developer

B. The Tester (QC)

C. Architectural designer

D. All the answers above.

3. What kind of Requirements does a Test Method need to follow?

A. The method must be decorated with the [TestMethod] attribute.

B. The method must return void.

C. The method cannot have parameters.

D. All the answers above.

4. What is the basic concept in Unit Testing?

A. The basic concept in Unit Testing is to Compare the results when running the Methods with some Input Data (“Actual”) with some Known Results (“Expected”) e.g: Assert.AreEqual(expected, actual, 0.001, ”Test failed because...");

B. Unit test can access the network resources, databases, file system, etc.

C. Unit testing must run complex.

D. All the answers above.

5. When should you write Unit Tests?

A. After you start coding or in parallel with the coding.

B. Before analyzing the system design.

C. Before you start coding or in parallel with the coding.

D. Unit testing is basically done after integration

6. What is the basic concept in Integration Testing?

A. Integration testing follows two approach known as ‘Top Down’ approach and ‘Bottom Up’ approach.

B. Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.

C. Integration testing is done by a specific integration tester or test team.

D. All the answers above.

7. What is the ‘Top Down’ approach

A. All components or modules are integrated simultaneously, after which everything is tested as a whole.

B. Testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers.

C. Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu).

D. All the answers above.

8. Which of the following answers about BDD is true?

A. BDD allows both the developer and the customer to work together to on requirements analysis that is contained within the source code of the system.

B. BDD focuses on the behavioural aspect of the system rather than the implementation aspect of the system that TDD focuses on.

C. BDD gives a clearer understanding as to what the system should do from the perspective of the developer and the customer. TDD only gives the developer an understanding of what the system should do.

D. All the answers above.

9. Refactoring is not one of the stages of TDD?

A. True

B. False

10. TDD required a developer to write the test cases before writing the actual production code.

A. True

B. False

11. TDD is basicaly Unit Testing?

A. True

B. False

12. TDD can help a developer improve his initial design of a method by forcing him to come up with situations in which the test case can fail.

    A. True

    B. False

13. TDD does not help in detection of bugs at an early developmental stages of SDLC.

    A. True

    B. False

14. There are _____ types of TDD?

    A. 1

    B. 2

    C. 3

    D. 4

15. TDD involves automating of test cases according to the need of the user?

    A. True

    B. False

Estimated Time to complete: 15 mins

Unit 4: Unit testing, Integration testing and Code quality control

Assignment 2: Unit Testing

Objectives:

Understanding about unit testing. Can using the Mocha, ChaiJS assertion, expectation to do the unit test.

Problem Descriptions:

As a product owner, I want to have a class simulation in Calculator. In the first sprint, I just want to have a function is divide().

The function divide() have two arguments are input1 and input2, when I use this function, I will receive the result is quotient of input1 / input2;

As a Developer, Please help me to write the test to make sure that, after the Class Calculator is given to test, it will be running true with the rules of math.

Please create test using Mocha framework with ChaiJS Assertion.

Technical Requirements:

- Using Mocha, ChaiJs.

Questions to answer:

2. There are at least 5 meaningful test cases tested.
3. Tested with inputs as: Numbers, numbers in string form, negative numbers, and division by 0.
4. The errors must be handled by a meaningful string.

Estimated Time to complete: 180 mins

Unit 4: Unit testing, Integration testing and Code quality control

Assignment 3: Integration testing with postman

Objectives:

Understanding about integration testing. Can using the postman to do the integration testing.

Problem Descriptions:

As a developer, I want to have a postman collection to test the API https://catfact.ninja

Technical Requirements:

- Using Postman.

Questions to answer:

1. There are at least 3 meaningful test cases tested on 3 endpoints.
2. The parameters must be generated by using the pre-script. The value must be stored in the variables.
3. The response must be expected by using the schema.

Estimated Time to complete: 180 mins

Unit 4: Unit testing, Integration testing and Code quality control

Assignment 4: Unit testing applying the TDD modal.

Objectives:

Understanding about unit testing applying the TDD modal. Can using mock, nock to fake request to DB or API.

Problem Descriptions:

As a developer, I want to implement a login function in the User class. This method allows two arguments, username and password. In this function, it calls a function to handle password encryption. Next, it calls a get function of the database library to look up the record with the username and encrypted password parameters.

Technical Requirements:

- Using Mocha, chaijs assertion, expectation and sinonjs to mock, fake the db library object. NYC coverage

Questions to answer:

1. All test case must have the meaningful. The test script must be successfully without connected to the database.
2. The code coverage must be 100%.
3. The DB library must be mocked by using the sinon mock(), fake() or stub().
4. The errors must be handled by a meaningful string.

Estimated Time to complete: 180 mins

-- THE END --